Porazdeljeni sistemi

Kaj so porazdeljeni sistemi?

- Več programskih komponent, ki tečejo na več računalnikih kot en sam (porazdeljen) sistem
- Računalniki so med seboj neodvisni
- Ti računalniki nasproti uporabnikom delujejo kot en sam koherenten sistem
- Lahko so to različni računalniki z različnimi OS
- Lahko so geografsko razpršeni in zelo oddaljeni
- Izvajajo skupna opravila

Cilji porazdeljenih sistemov?

- Uporabljamo jih za povečevanje
  - zmogljivosti
  - zanesljivosti
- Povezovanje porazdeljenih virov:
  - transparentno (nevidno za uporabnika)
  - odprto (zagotavlja povezovanje med različnimi platformami)
  - skalabilno (povečevanja števila uporabnikov ali virov)
  - robustno (neobčutljivost za napake npr. v omrežju, izpadov računalnikov...)
  - varno (preprečevanje morebitnih vdorov in zlorab)

Organizacija porazdeljenih sistemov

- Opravka imamo z različnimi arhitekturami
  - protokoli komunikacije morajo biti neodvisni od platforme
- Porazdeljenost
  - skrb za nepravilno dostavljena, neveljavna sporočila, robustnost
- Prenosljivost programske opreme
  - možnost prenosa in poganjanje programske opreme med različnimi strojnimi in programskimi platformami

Transparentnost v porazdeljenih sistemih

- Skrivanje porazdeljenosti pred uporabniki
- Različne vrste transparentnosti/skrivanja:
  - dostop: dostop do virov mora potekati na en sam način, neodvisno od načina dostopa na posameznem računalniku
  - lokacija: fizična lokacija vira
  - migracija: možnost selitve vira iz ene lokacije na drugo
  - relokacija: selitev vira med samo uporabo
  - replikacija: vir se replicira na več lokacijah
  - sočasnost: vir uporablja več uporabnikov hkrati
  - napake: pri dostopu do vira prihaja do napak in reševanja napak
  - trajnost: začasnost ali trajnost dostopnosti vira
  - varnost: tehnike varnosti in zaščite uporabljene pri dostopu naj bi bile čimbolj skrite
- Sistemi seveda ne dosegajo popolne transparentnosti pri vseh naštetih vrstah – cilj je čimvečja transparentnost

Odprtost v porazdeljenih sistemih

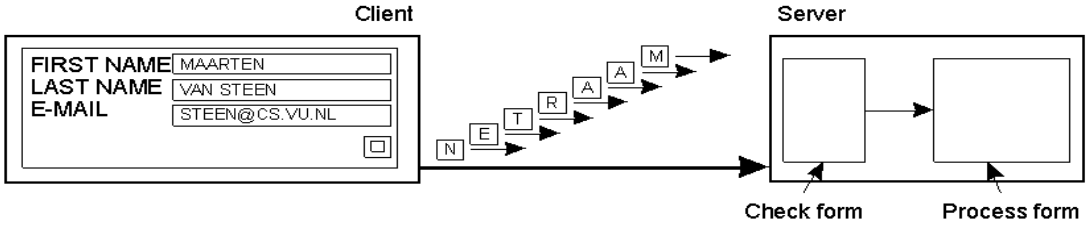
- Odprtost zagotavlja povezovanje med različnimi platformami
- Odprti sistemi so bolj skalabilni kot zaprti sistemi
  - povezovanje med več platformami
- Protokoli komunikacije in izmenjave podatkov morajo biti neodvisni od platforme
  - npr. spletne storitve (web services)
    - HTTP kot protokol izmenjave podatkov
    - XML kot standard za zapis podatkov

Skalabilnost v porazdeljenih sistemih

- Skalabilen je sistem, ki ga lahko enostavno spremenimo tako, da bo dovoljeval povečevanje števila uporabnikov, virov in računalnikov
- Dimenzije skalabilnosti:
  - obremenitev: porazdeljen sistem mora dovoljevati enostavno povečevanje/zmanjševanje števila virov, da zadosti večji ali manjši obremenitvi
  - geografska: ohranjanje uporabnosti in odzivnosti ne glede na oddaljenost uporabnikov in virov
  - administrativna: enostavnost administracije naj ne bi bila večja z večanjem števila uporabnikov/organizacij
- Porazdeljeni sistemi porabljajo del svojih virov za zagotavljanje skalabilnosti
  - obstaja omejitev ,do katere lahko povečujemo obremenitev v eni od naštetih dimenzij, preden se performanse začnejo slabšati

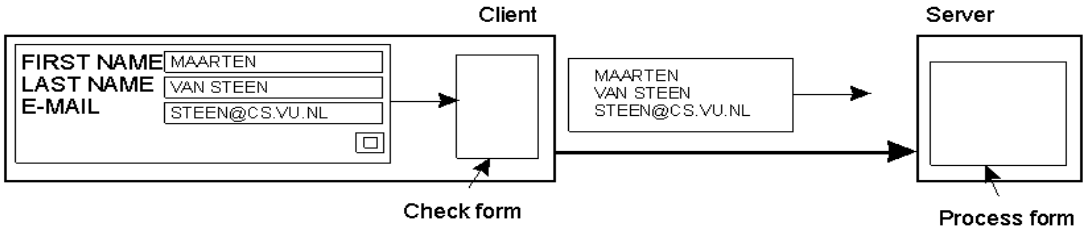
Kako dosegamo skalabilnost (1)

- Primer: centralizirane storitve
  - sistem odjemalec-strežnik, odjemalci so “tanki” (thin-client)
  - vse delo opravlja strežnik, vsi odjemalci se povezujejo z njim
  - slaba skalabilnost glede na povečevanje števila uporabnikov, saj vsi močno obremenjujejo strežniške vire



(a)

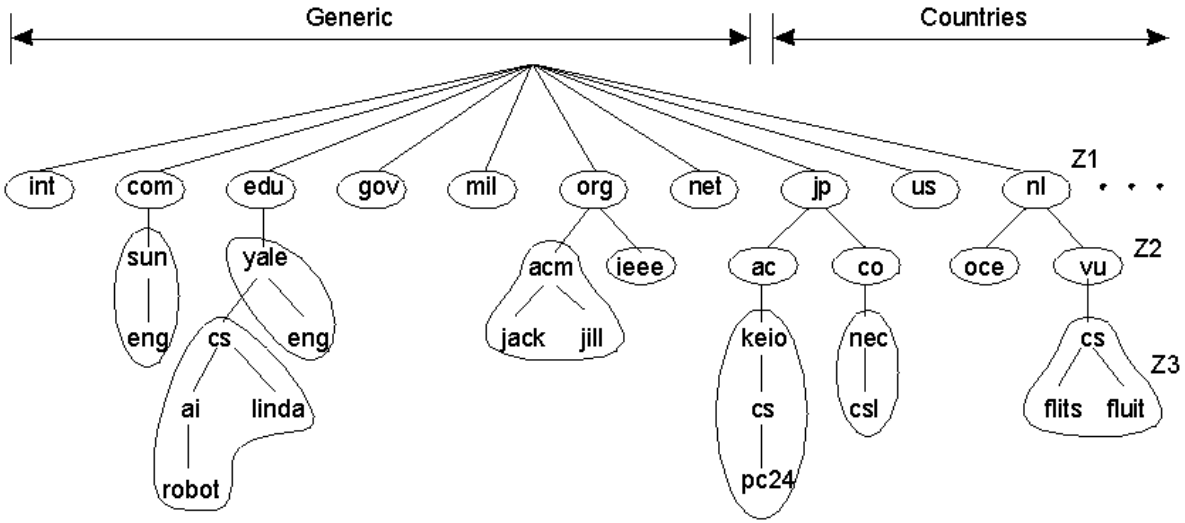
- Rešitev:
  - del opravil prestavimo na odjemalca, zmanjšamo centralizacijo
  - posledično zmanjšamo obremenitev strežnika
  - če je koda portabilna (npr. Javascript), lahko ohranimo odprtost



(b)

Kako dosegamo skalabilnost (2)

- Primer: centralizirani podatki
  - strežnik vsebuje vse podatke
  - vsi odjemalci morajo komunicirati s tem edinim strežnikom, da pridejo to teh podatkov
  - slaba skalabilnost, saj morajo vsi odjemalci dostopati do tega strežnika
- Rešitev:
  - podatke porazdelimo po več strežnikih
  - lep primer: strežniki DNS (Domain Name Servers)



Kako dosegamo skalabilnost (3)

- Primer: centralizirani algoritmi
  - algoritem pozna celotno stanje sistema
  - vrstni red sprememb stanj in dogodkov je znan
  - primer: upodabljanje (rendering) animacije:
    - for slika=1..n
      - naloži sceno
      - for del slike=1..m
        - » upodobi del slike (n,m)
  - upodabljanje je zaporedno, čas dela je reda nxm
- Povečanje skalabilnosti s porazdeljenim algoritmom
  - posamezne enote ne poznajo celotnega stanja (v zgornjem primeru npr. kaj je že upodobljeno, niti nujno celotne scene)
  - vrstni red upodabljanj delov scene ni vnaprej znan
  - primer algoritma na eni enoti:
    - ponavljaj dokler ne zmanjka delov scen
      - zahtevaj naslednji del scene za upodabljanje
      - upodobi ta del scene

Robustnost (fault tolerance)

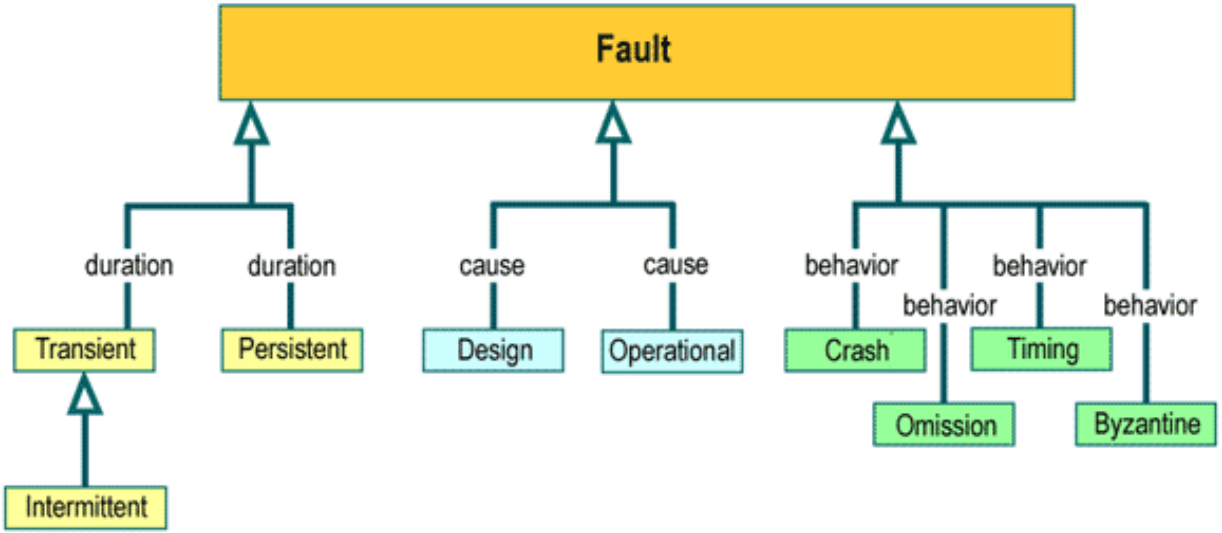
- je zmožnost sistema, da deluje pravilno tudi v primeru notranjih napak.
  - npr. odpoved računalnika, dela omrežja...
- povečanje robustnosti pomeni, da povečamo zanesljivost sistema
- Definicije:
  - odpoved (failure) nastane, ko sistem ne deluje več tako, kot bi moral (ni v skladu s specifikacijo)
  - vzrok odpovedi je napačno stanje (error) sistema, torej stanje, ki ni predvideno v specifikaciji
    - napačno stanje lahko povzroči nič ali več odpovedi
  - vzrok napačnega stanja je napaka (fault), ki je osnovni vzrok odpovedi.
    - napaka lahko povzroči nič ali več napačnih stanj

Robustnost (fault tolerance)

- Primer:
  - v programu nepravilno povečamo vrednost neke spremenljivke, namesto da bi jo zmanjšali. Če se ta stavek izvede, bo spremenljivka imela napačno vrednost
  - če drugi programski stavki uporabljajo to vrednost, lahko sistem deluje nepravilno
    - napačni stavek je napaka
    - napačna vrednost spremenljivke je napačno stanje
    - odpoved je vedenje sistema po tem, ko je prišlo do napačnega stanja
  - če vrednosti spremenljivke (napačnega stanja) ostali stavki ne uporabljajo, ali če se stavek (napaka) sploh ne izvrši, do odpovedi ne bo prišlo

Robustnost (fault tolerance)

- Klasifikacija napak:
  - po trajanju: prehodne (tudi ponavljajoče) in stalne
  - po vzroku: posledica načrtovanja ali delovanja
  - po vedenju:
    - sesutje – komponenta preneha z delovanjem
    - izpustitvene napake – komponenta ne izvede naloge
    - zakasnitvene napake – komponenta ne izvede naloge pravočasno
    - bizantinske napake – napake največkrat zlonamerno povzročene



Napake pri porazdeljenih sistemih

- Odpoved računalnika
  - ostali računalniki v sistemu morajo odpoved zaznati in pravilno ukrepati, npr. s porazdelitvijo opravil na druge računalnike
- Odpoved komunikacijskih povezav
  - če odpove omrežje, sistem razpade na posamezne kose, ki lahko posledično delujejo neusklajeno in nasprotujoče
  - problemi pri prenosu sporočil (izguba, podvojevanje, vrstni red)
  - rešujemo s:
    - protokoli za potrditev prejema
    - omejen največji dovoljen čas od poslanega sporočila do prejema potrditve – če potrditev ne pride, sporočilo pošljemo ponovno
- Zakasnitve
  - velike zakasnitve pri izmenjavi sporočil lahko npr. vodijo do napačnih informacij o stanju sistema (komponenta je že v drugem stanju, ko je sporočilo prejeto)
  - vrstni red sporočil je lahko drugačen v različnih delih sistema
  - reševanje z algoritmi za porazdeljeno sporazumevanje (distributed agreement). V kategorijo sodijo:
    - časovna sinhronizacija, vodenje stanja, porazdeljeni semaforji, porazdeljene transakcije ...

Napake pri porazdeljenih sistemih

- Standardne napačne predpostavke pri razvoju porazdeljenih sistemov
  - omrežje je zanesljivo
  - zakasnitev ni
  - pretok je neskončen
  - omrežje je varno
  - topologija se ne spreminja
  - obstaja samo en administrator
  - prenos podatkov je zastonj

Tipi porazdeljenih sistemov

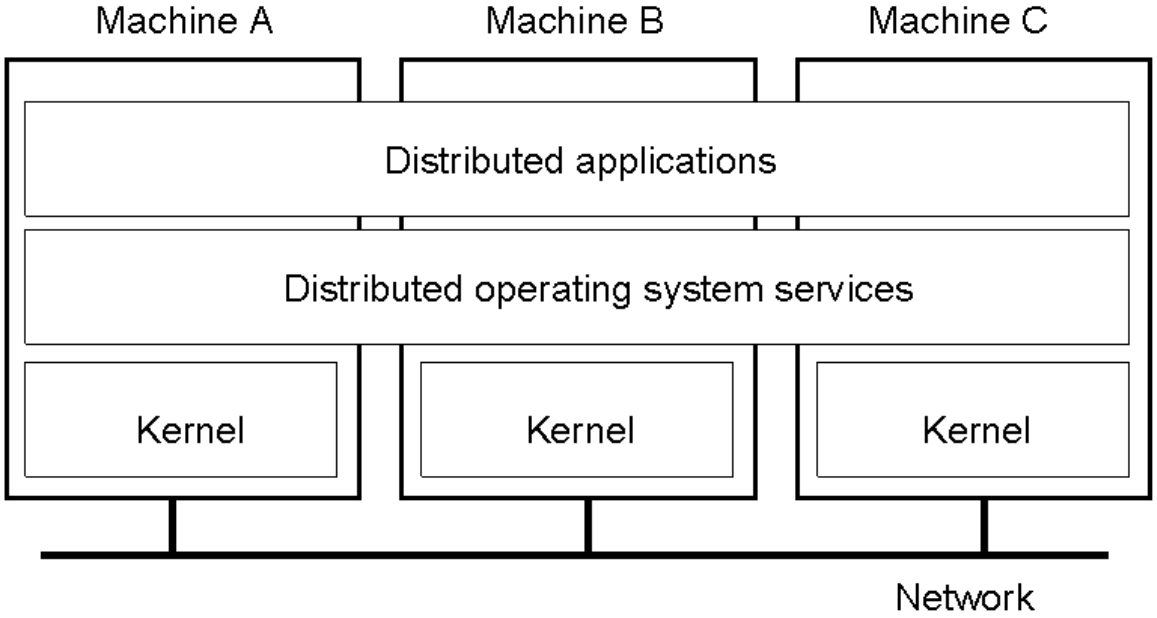
- Multiprocesorski sistemi
  - računalnik z več CPE, deljen pomnilnik
  - zadnje čase tudi same CPE podpirajo multiprocesiranje (Intel Core 2, Athlon 64 X2 ...)
- Multiračunalniški sistemi: več računalnikov v omrežju
  - homogeni: enake CPE, en tip omrežja
    - npr. gruče (clusters) navadno sodijo v to kategorijo
  - heterogeni: različni računalniki, omrežja ..
    - npr. omrežja grid povezujejo velike količine računalnikov preko interneta

OS pri porazdeljenih sistemih

Sistem	Opis	Glavni cilj
MOS – multiprocessor OS – multiprocesorski OS	tesno sklopljen OS za multiprocesorje komunikacija preko deljenega pomnilnika navzven daje vtis enega sistema	transparentnost upravljanja z viri
DOS – distributed OS - porazdeljeni OS	OS za multiprocesorske in multiračunalniške arhitekture. navzven daje vtis enega sistema	nudi transparentnost upravljanja s porazdeljenimi viri
NOS – network OS - omrežni OS	rahlo sklopljen OS za heterogene porazdeljene sisteme navzven ne daje vtisa enega sistema	ponuja lokalne storitve oddaljenim odjemalcem
Middleware	dodatni nivo nad NOS, ki nudi storitve, ki ga približajo DOS	nudi transparentnost pri porazdeljenih storitvah

Porazdeljeni OS

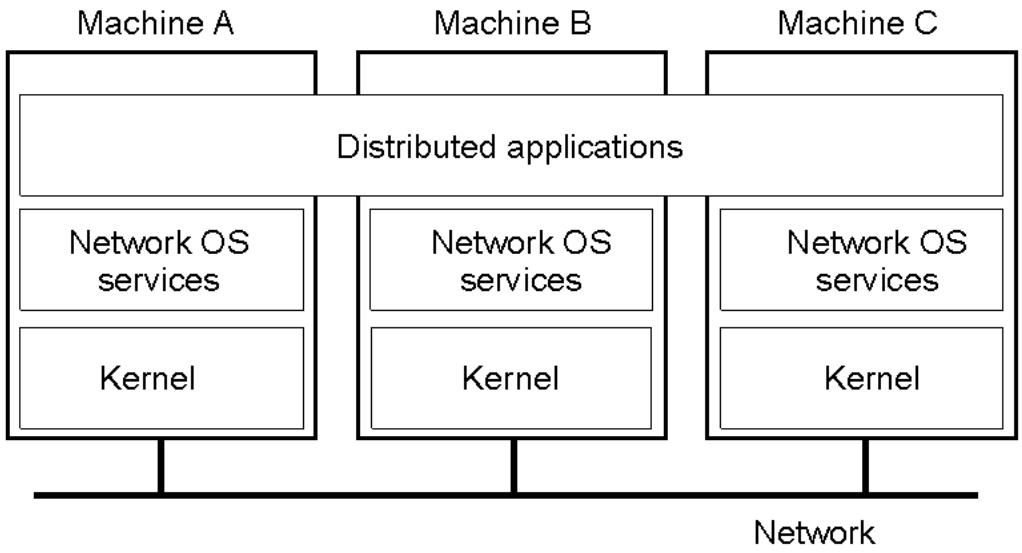
- Struktura multiračunalniškega DOS



- Na nivoju OS rešuje:
  - transparentnost (lokacija, replikacija, sočasnost ...)
    - npr. porazdeljeni datotečni sistemi (DFS)
  - upravljanje z viri za optimizacijo performans
  - skalabilnost
  - zanesljivost

Omrežni OS

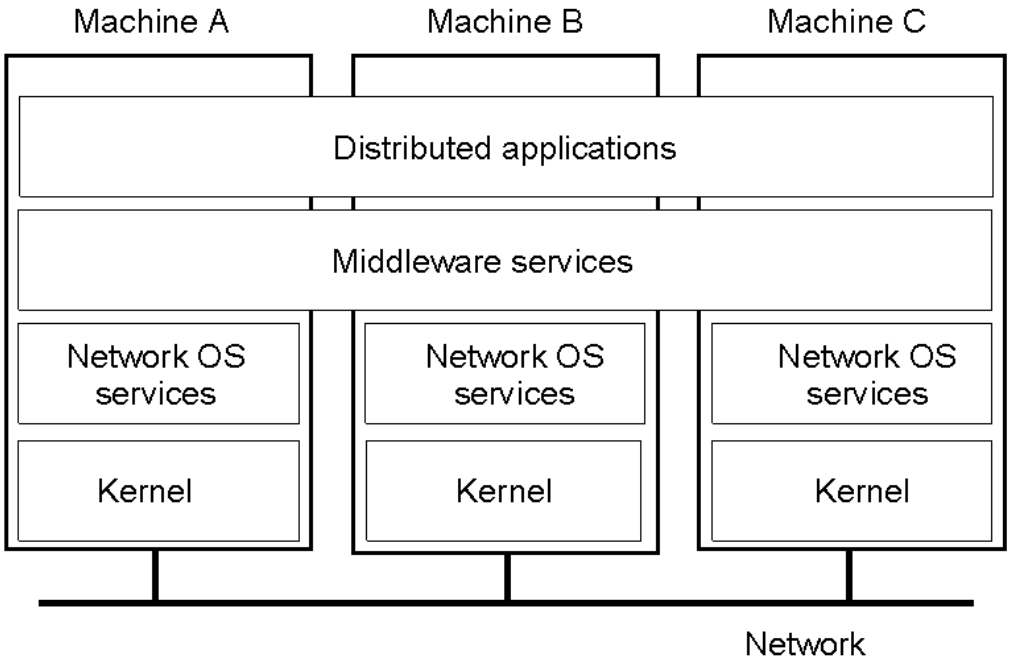
- Struktura omrežnega OS.



- Poleg standardnih storitev, OS nudi še omrežne storitve, ki jih uporabljajo porazdeljene aplikacije
  - deljenje datotek in drugih virov, RPC, web, ftp ...
- Odjemalci morajo eksplicitno vedeti kateri računalniki nudijo posamezne storitve
- V to kategorijo spada večina današnjih OS (Windows Server 2003, Linux ...)

Porazdeljene aplikacije in omrežni OS

- Problemi pri omrežnih OS:
  - sami ne rešujejo transparentnosti, upravljanja z viri, skalabilnosti, itn.
  - le nudijo omrežne servise
- Za implementacijo porazdeljenih sistemov nad NOS je smiselno uporabljati vmesni nivo: middleware

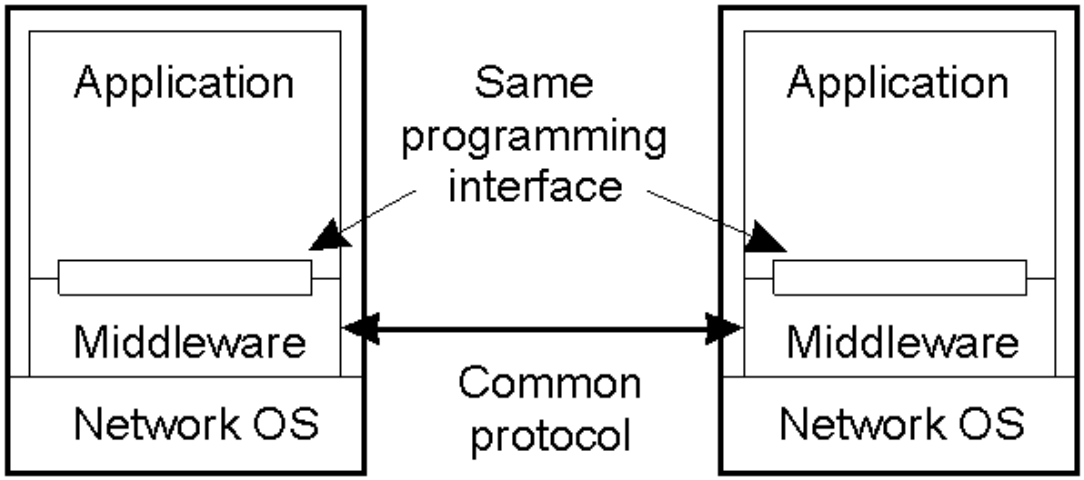


Middleware

- Middleware nudi storitve nad NOS, ki omogočajo lažji razvoj porazdeljenih aplikacij:
  - komunikacija med procesi
  - poimenovanje virov
  - trajanje virov
  - porazdeljene transakcije
  - varnost
- Modeli, ki jih middleware uporablja, so npr.:
  - porazdeljeni datotečni sistemi – DFS Distributed File System
  - klici oddaljenih procedur - RPC Remote Procedure Call
  - porazdeljeni objekti
  - porazdeljeni dokumenti

Middleware in odprtost

- Middleware povezuje heterogene sisteme
  - posledično mora biti odprt in grajen po standardih
- Aplikacije, ki uporabljajo middleware, uporabljajo enoten API (programming interface)
- Komponente middleware-a uporabljajo skupne protokole za komunikacijo



Primerjava med OS

	Multiproc. OS	**Porazdeljeni OS**	**Omrežni OS**	**Omrežni OS + Middleware**
Transparentnost	zelo visoka	visoka	nizka	visoka
Enak OS na vseh vozliščih	da	da	ne	ne
Število kopij OS	1	n	n	n
Osnova za komunikacijo	deljeni pomnilnik	sporočila	datoteke	odvisno od modela
Upravljanje z viri	globalno, centralno	globalno, porazdeljeno	vsako vozlišče svoje vire	po vozliščih
Skalabilnost	ne	srednja	da	odvisno od modela
odprtost	zaprt	zaprt	odprt	odprt